

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Atsushi MASUDA

GAU:

SERIAL NO: New Application

EXAMINER:

FILED: Herewith

FOR: DESIGN METHOD OF LOGIC CIRCUIT

REQUEST FOR PRIORITY

COMMISSIONER FOR PATENTS
ALEXANDRIA, VIRGINIA 22313

SIR:

Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of 35 U.S.C. §120.

Full benefit of the filing date(s) of U.S. Provisional Application(s) is claimed pursuant to the provisions of 35 U.S.C. §119(e): Application No. Date Filed

Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of 35 U.S.C. §119, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
Japan	2002-354208	December 5, 2002

Certified copies of the corresponding Convention Application(s)

are submitted herewith

will be submitted prior to payment of the Final Fee

were filed in prior application Serial No. filed

were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

(A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

(B) Application Serial No.(s)
 are submitted herewith
 will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.



Marvin J. Spivak

Registration No. 24,913

C. Irvin McClelland
Registration Number 21,124



22850

Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 05/03)

日本特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日

Date of Application:

2002年12月 5日

出願番号

Application Number:

特願2002-354208

[ST.10/C]:

[JP2002-354208]

出願人

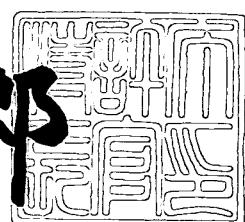
Applicant(s):

株式会社東芝

2002年12月27日

特許庁長官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2002-3102614

【書類名】 特許願
【整理番号】 ASB023110
【提出日】 平成14年12月 5日
【あて先】 特許庁長官殿
【国際特許分類】 G06F 17/50
【発明の名称】 論理回路設計方法、論理回路設計プログラムおよび論理回路設計装置
【請求項の数】 15
【発明者】
【住所又は居所】 神奈川県川崎市幸区小向東芝町1番地 株式会社東芝マ
イクロエレクトロニクスセンター内
【氏名】 増田 篤司
【特許出願人】
【識別番号】 000003078
【氏名又は名称】 株式会社 東芝
【代理人】
【識別番号】 100083806
【弁理士】
【氏名又は名称】 三好 秀和
【電話番号】 03-3504-3075
【選任した代理人】
【識別番号】 100068342
【弁理士】
【氏名又は名称】 三好 保男
【選任した代理人】
【識別番号】 100100712
【弁理士】
【氏名又は名称】 岩▲崎▼ 幸邦

【選任した代理人】

【識別番号】 100100929

【弁理士】

【氏名又は名称】 川又 澄雄

【選任した代理人】

【識別番号】 100108707

【弁理士】

【氏名又は名称】 中村 友之

【選任した代理人】

【識別番号】 100095500

【弁理士】

【氏名又は名称】 伊藤 正和

【選任した代理人】

【識別番号】 100101247

【弁理士】

【氏名又は名称】 高橋 俊一

【選任した代理人】

【識別番号】 100098327

【弁理士】

【氏名又は名称】 高松 俊雄

【手数料の表示】

【予納台帳番号】 001982

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 論理回路設計方法、論理回路設計プログラムおよび論理回路設計装置

【特許請求の範囲】

【請求項1】 論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変換するステップと、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入するステップと、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータバスと、前記データバスの制御情報を生成するステップと、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成するステップとを有することを特徴とする論理回路設計方法。

【請求項2】 前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成するステップと、

前記データフロー図と前記データバスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成するステップをさらに有し、

前記演算子演算器データベースを生成するステップが、前記ノード演算子データベースと前記ノード演算器データベースに基づくことを特徴とする請求項1に記載の論理回路設計方法。

【請求項3】 前記データバスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成するステップと、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算するステップと、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算するステップと、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索するステップと、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定するステップとをさらに有することを特徴とする請求項1または請求項2に記載の論理回路設計方法。

【請求項4】 演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索するステップをさらに有することを特徴とする請求項1乃至請求項3のいずれか1つに記載の論理回路設計方法。

【請求項5】 予想される全実行時間が最短になるようにデータフロー図を変更するステップとをさらに有することを特徴とする請求項1乃至請求項4のいずれか1つに記載の論理回路設計方法。

【請求項6】 論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する手順と、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入する手順と、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータバスと、前記データバスの制御情報を生成する手順と、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成する手順をコンピュータに実行させるための論理回路設計プログラム。

【請求項7】 前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが

記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成する手順と、

前記データフロー図と前記データパスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成する手順をさらにコンピュータに実行させるためにあり

、
前記演算子演算器データベースを生成する手順が、前記ノード演算子データベースと前記ノード演算器データベースに基づくこととを特徴とする請求項6に記載の論理回路設計プログラム。

【請求項8】 前記データパスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成する手順と、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算する手順と、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算する手順と、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索する手順と、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定する手順をさらにコンピュータに実行させるための請求項6または請求項7に記載の論理回路設計プログラム。

【請求項9】 演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索する手順をさらにコンピュータに実行させるための請求項6乃至請求項8のいずれか1つに記載の論理回路設計プログラム。

【請求項10】 予想される全実行時間が最短になるようにデータフロー図を変更する手順をさらにコンピュータに実行させるための請求項6乃至請求項9のいずれか1つに記載の論理回路設計プログラム。

【請求項11】 論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変

換する構文解析部と、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入するスケジューリング部と、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータパスと、前記データパスの制御情報を生成するハードウェア割付部と、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成する演算子演算器対応情報解析部を有することを特徴とする論理回路設計装置。

【請求項12】 前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成するノード演算子対応情報解析部と、

前記データフロー図と前記データパスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成するノード演算器対応情報解析部をさらに有し、

前記演算子演算器データベースを生成することが、前記ノード演算子データベースと前記ノード演算器データベースに基づくことを特徴とする請求項11に記載の論理回路設計装置。

【請求項13】 前記データパスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成するレジスタ転送レベル記述生成部と、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算する動作記述シミュレーション部と、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算するレジスタ転送レベルシミュレーシ

ヨン部と、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索する演算子検索部と、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定する記憶データ判定部をさらに有することを特徴とする請求項11または請求項12に記載の論理回路設計装置。

【請求項14】 演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索する演算器検索部をさらに有することを特徴とする請求項11乃至請求項13のいずれか1つに記載の論理回路設計装置。

【請求項15】 予想される全実行時間が最短になるようにデータフロー図を変更する最適化部をさらに有することを特徴とする請求項11乃至請求項14のいずれか1つに記載の論理回路設計装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、レジスタ転送レベル（RTL）記述を自動生成する高位合成技術に関し、特に、動作記述からRTL記述に変換する論理回路設計方法に関する。

【0002】

【従来の技術】

従来、論理回路設計では、動作記述の中間信号と論理回路の演算器との対応表が出力された。（例えば、特許文献1参照。）。

【0003】

しかし、さらに高位合成されたRTL記述では、実行ステップ毎にデータが記憶素子に記憶され、1つの演算器が繰り返し演算を行う。このため、対応表は存在せず、実行ステップ毎の対応表も存在しなかった。RTL記述の論理回路のシミュレーション結果に不具合があった場合は、設計者がRTL記述のみを用いてそのシミュレーション結果を解析した。RTL記述は、動作時間や条件などにより動作する部分が変わるので、この解析には非常に多くの時間を要した。

【0004】

【特許文献1】

特開平11-73447号公報（請求項1、図6）

【0005】

【発明が解決しようとする課題】

本発明は、上記事情に鑑みてなされたものであり、その目的とするところは、動作記述から変換されたRTL記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供することにある。

【0006】

本発明の目的は、コンピュータが動作記述から変換されたRTL記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計プログラムを提供することにある。

【0007】

本発明の目的は、動作記述から変換されたRTL記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計装置を提供することにある。

【0008】

【課題を解決するための手段】

上記問題点を解決するための本発明の第1の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換すること、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入すること、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成すること、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成することとを有する論理回路設計方法にある。

【0009】

本発明の第2の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動

作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する手順と、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する手順と、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成する手順と、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する手順をコンピュータに実行させるための論理回路設計プログラムにある。

【0010】

本発明の第3の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する構文解析部と、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入するスケジューリング部と、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成するハードウェア割付部と、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する演算子演算器対応情報解析部を有する論理回路設計装置にある。

【0011】

【発明の実施の形態】

次に、図面を参照して、本発明の実施の形態について説明する。以下の図面の記載において、同一又は類似の部分には同一又は類似の符号を付している。また、図面は模式的なものであり、現実のものとは異なることに留意すべきである。

【0012】

(論理回路設計装置)

本発明の実施の形態に係る論理回路設計装置1は、図1に示すように、構文解析部2、スケジューリング部3、最適化部4、ハードウェア割付部5、RTL記述生成部6、ノード演算子対応情報解析部7、ノード演算器対応情報解析部8、演算子演算器対応情報解析部9、動作記述シミュレーション部10、RTL記述シミュレーション部11、シミュレーション結果比較部12と入出力部16を有している。シミュレーション結果比較部12は、演算子検索部13、記憶データ判定部14と演算器検索部15を有している。

【0013】

構文解析部2は、図2に示すように、論理回路のアルゴリズムを、複数の演算子を有する動作記述D1を入力し、演算子を実行する演算ノードを実行順に配列した未加工データフロー図(DFG)D2を出力する。

【0014】

スケジューリング部3は、未加工データフロー図D2を入力し、未加工データフロー図D2に実行ステップを割り当てる。スケジューリング部3は、演算ノードからの出力データを記憶するレジスタを、未加工データフロー図D2の実行ステップの実行後に挿入する。スケジューリング部3は、スケジューリングデータフロー図D3を出力する。

【0015】

最適化部4は、スケジューリングデータフロー図D3を入力する。最適化部4は、論理回路のアルゴリズムの予想される全実行時間が最短になるように、スケジューリングデータフロー図D3を変更し、最適化データフロー図D4を出力する。

【0016】

ハードウェア割付部5は、最適化データフロー図D4を入力する。ハードウェア部5は、演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスD5と、データパスの制御情報D5を生成する。

【0017】

RTL記述生成部6は、データパスと制御情報D5を入力する。RTL記述生

成部6は、データパスと制御情報D5に基づいて、論理回路のRTL記述D6を生成する。RTL記述生成部6は、RTL記述D6を出力する。

【0018】

ノード演算子対応情報解析部7は、動作記述D1、スケジューリングデータフロー図D3と最適化データフロー図D4を入力する。ノード演算子対応情報解析部7は、動作記述D1、スケジューリングデータフロー図D3と最適化データフロー図D4に基づいて、ノード演算子データベースD7を生成する。ノード演算子データベースD7は、演算ノードで実行する演算子を演算ノードから検索可能であり、レジスタが記憶するデータを出力する演算ノードで実行する演算子をレジスタから検索可能である。

【0019】

ノード演算器対応情報解析部8は、最適化データフロー図D4とデータパスD5を入力する。ノード演算器対応情報解析部8は、最適化データフロー図D4とデータパスD5に基づいて、ノード演算器データベースD8を生成する。ノード演算器データベースD8は、演算ノードとして機能する演算器を演算ノードから検索可能であり、記憶素子の機能をするレジスタを実行ステップと記憶素子から検索可能である。

【0020】

演算子演算器対応情報解析部9は、ノード演算子データベースD7とノード演算器データベースD8を入力する。演算子演算器対応情報解析部9は、ノード演算子データベースD7とノード演算器データベースD8に基づいて、演算子演算器データベースD9を生成する。演算子演算器データベースD9は、演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である。

【0021】

動作記述シミュレーション部10は、動作記述D1と入力データD13を入力する。動作記述シミュレーション部10は、動作記述D1に入力データD13を代入し、演算子からの出力データD10を計算する。

【0022】

R T L記述シミュレーション部11は、R T L記述D6と入力データD13を入力する。R T L記述シミュレーション部11は、RTL記述D6に入力データD13を代入し、実行ステップ毎に記憶素子が記憶する記憶データD11を計算する。

【0023】

シミュレーション結果比較部12の演算子検索部13は、演算子演算器データベースD9と、記憶データD11を記憶した際の実行ステップと記憶素子を入力する。演算子検索部13は、演算子演算器データベースD9に基づいて、計算された記憶データの実行ステップと記憶素子から演算子を検索する。

【0024】

シミュレーション結果比較部12の記憶データ判定部14は、記憶データD11と、検索された演算子の出力データを入力する。記憶データ判定部14は、検索された演算子の出力データと記憶データの異同を判定する。検索された演算子の出力データと記憶データが異なる場合は、R T L記述D6に不具合があると判定する。この判定に基づいて、R T L記述D6の不具合の有無D12を出力する。R T L記述D6に不具合がある場合は、さらに、記憶データD11を記憶した際の実行ステップ、記憶素子と演算子D12を出力する。

【0025】

シミュレーション結果比較部12の演算器検索部15は、検索された演算子の出力データと記憶データが異なる場合に、演算子演算器データベースD9と検索された演算子を入力する。演算器検索部15は、演算子演算器データベースD9に基づいて、検索された演算子から演算器を検索する。演算器検索部15は、検索された演算器D12を出力する。

【0026】

入出力部16は、動作記述D1と入力データD13を入力する。入出力部16は、R T L記述D6とR T L記述D6の不具合の有無、不具合の発生した実行ステップ、記憶素子、演算子と演算器D12を出力する。

【0027】

高位合成でC記述等の動作記述D1からRTL記述D6を生成する際に、動作記述

D1とRTL記述D6の両者を対応づける演算子演算器データベースD9を生成する。RTL記述D6の検証に、動作記述D1の検証に用いたテストベクトルを使うことができ、かつ、動作記述D1との比較検証をすることにより、不具合が生じた場合には、動作記述D1、RTL記述D6の両者での不具合箇所が特定できる。このことにより、動作記述D1やRTL記述D6の修正が容易に行えるようになり、設計、検証、不具合修正にかかる時間を大幅に短縮することができる。

【0028】

(論理回路設計方法)

本発明の実施の形態に係る論理回路設計方法は、図3に示すように、ステップS1において、構文解析部2で、構文解析をする。論理回路のアルゴリズムを、複数の演算子を有する動作記述D1から、演算子を実行する演算ノードを実行順に配列した未加工データフロー図(DFG)D2に変換する。

【0029】

ステップS2において、スケジューリング部3で、スケジューリングをする。未加工データフロー図D2に実行ステップを割り当て、演算ノードにどの実行ステップで実行するかという時間情報が添付される。未加工データフロー図D2に、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する。このレジスタにより、実行ステップをまたがるデータを保持することができる。スケジューリング部3は、このように加工された未加工データフロー図D2をスケジューリングデータフロー図D3として出力する。

【0030】

ステップS3において、最適化部4で、予想される全実行時間が最短になるようにスケジューリングデータフロー図D3を変更する。このように変更されたスケジューリングデータフロー図D3を最適化データフロー図D4として出力する。

【0031】

ステップS4において、ハードウェア割付部5で、演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスとこのデータパスの制御情報D5を生成する。

【0032】

ステップS5において、RTL記述生成部6で、データパスと制御情報D5に基づいて、RTL記述D6を生成する。

【0033】

ステップS6において、RTL記述シミュレーション部11で、実行ステップiに、最初に実行される実行ステップ1を設定する。

【0034】

ステップS7において、RTL記述シミュレーション部11で、RTL記述D6のシミュレーションを行う。RTL記述D6に入力データD13を代入し、実行ステップiで演算器が出力するデータと記憶素子が記憶する記憶データD11を計算する。

【0035】

一方、ステップS2とS3の実行後に、ステップS8において、ノード演算子対応情報解析部7で、動作記述D1とスケジューリングデータフロー図D3と最適化データフロー図D4に基づいて、ノード演算子データベースD7を生成する。

【0036】

ステップS4の実行後に、ステップS9において、ノード演算器対応情報解析部8で、最適化データフロー図D4とデータパスD5に基づいて、ノード演算器データベースD8を生成する。

【0037】

ステップS10において、演算子演算器対応情報解析部9で、ノード演算子データベースD7とノード演算器データベースD8に基づいて、演算子演算器データベースD9を生成する。

【0038】

ステップS11において、動作記述シミュレーション部10で、動作記述D1のシミュレーションを行う。動作記述D1に入力データD13を代入し、演算子からの出力データD10を計算する。動作記述を実際にコンパイルし実行した場合の各演算子が出力する中間データを計算する。

【0039】

ステップS7とS11の実行後に、ステップS12において、シミュレーション結果比較部12の演算子検索部13で、演算子演算器データベースD9に基づいて、計算された記憶データD11の実行ステップiと記憶素子から演算子を検索する。

【0040】

ステップS13において、記憶データ判定部14で、検索された演算子の出力データD10と、記憶データD11の異同を判定する。異なるとの判定の場合は、ステップS16に進み、同じとの判定の場合は、ステップS14に進む。

【0041】

ステップS16において、演算器検索部15で、演算子演算器データベースD9に基づいて、検索された演算子から演算器を検索する。

【0042】

ステップS17において、論理回路設計装置1あるいはそのオペレータが、検索された演算器に基づいて、RTL記述D6をデバックする。ステップS5に戻る。なお、戻る先は、ステップS5に限らない。再度シミュレーションされるRTL記述がデバックされたRTL記述であれば、ステップS6に戻ってもよい。

【0043】

一方、ステップS14において、シミュレーション結果比較部12で、実行ステップiが最適化データフロー図D4の最大の実行ステップ以上であるか否かを判定する。実行ステップiが最大の実行ステップ以上でない場合は、ステップS15に進む。ステップS15において、実行ステップiを1つ大きくする。そして、ステップS7に進む。実行ステップiが最大の実行ステップ以上である場合は、論理回路設計方法をストップする。このように、動作記述D1のミュレーション結果である演算子の出力データD10と、RTL記述D6のシミュレーション結果である記憶素子の記憶データD11を、実行ステップ毎に比較することで、動作記述と高位合成されたRTL記述との比較検証が行われる。

【0044】

動作記述D1とRTL記述の比較検証をすることにより、不具合が生じた場合に

は、動作記述D 1、RTL記述D 6の両者での不具合箇所が特定できる。このことにより、動作記述D 1やRTL記述D 6の修正が容易に行えるようになり、設計、検証、不具合修正にかかる時間を大幅に短縮することができる。

【0045】

尚、RTL記述D 6のシミュレーションでは、1実行ステップ毎にデータを比較しつつシミュレーションを行っている。これに限らず、RTL記述D 6の全実行ステップの実行後に、動作記述D 1とRTL記述の比較検証を行ってもよい。ただ、先の実行ステップでの不具合が、後の実行ステップでの不具合を起こす場合がある一方で、先の実行ステップでの不具合のデバックが、後の実行ステップで不具合を起こす場合もある。従って、動作記述D 1とRTL記述の比較検証の時期は、1実行ステップ毎か、全実行ステップの実行後かを、場合により使い分けることが望ましい。

【0046】

(実施例1)

実施例1では、論理回路設計装置1で論理回路設計方法を用いた。実施例1の論理回路設計方法は、図3のステップS 1において、図4に示すような複数の演算子 o_{p1} 乃至 o_{p4} を有する動作記述D 1から、図5に示すような演算子 o_{p1} 乃至 o_{p4} を実行する演算ノードN 1乃至N 4を実行順に配列した未加工データフロー図D 2に変換する。演算子 o_{p1} 乃至 o_{p4} は、実際には内部のデータとして管理され、識別番号やポインタ等で認識される。入力データD 1 3の引数 i_1 と i_2 で、演算ノードN 1において、和算の演算子 o_{p1} を実行する。引数 i_1 と i_2 で、演算ノードN 2において、引き算の演算子 o_{p3} を実行する。演算ノードN 1の演算子 o_{p1} の出力と演算ノードN 2の演算子 o_{p2} の出力で、演算ノードN 3において、積算の演算子 o_{p2} を実行する。演算ノードN 3の演算子 o_{p2} の出力と定数1で、演算ノードN 4において、和算の演算子 o_{p4} を実行する。

【0047】

ステップS 2において、図6に示すように、図5の未加工データフロー図D 2に実行ステップ1乃至3を割り当てる。演算ノードN 1とN 2は、実行ステップ

1で実行するという時間情報が添付される。演算ノードN3は、実行ステップ2で実行するという時間情報が添付される。演算ノードN4は、実行ステップ3で実行するという時間情報が添付される。未加工データフロー図D2に、演算ノードN1乃至N4からの出力データを記憶するレジスタR1乃至R4を実行ステップ1乃至3の実行後に挿入する。スケジューリング部3は、このように加工した図5の未加工データフロー図D2を、図6のスケジューリングデータフロー図D3として出力する。

【0048】

ステップS3において、スケジューリングデータフロー図D3を最適化するが、既に、予想される全実行時間が最短であるので、出力された最適化データフロー図D4は、スケジューリングデータフロー図D3に等しい。

【0049】

ステップS8において、図4の動作記述D1と図6のスケジューリングデータフロー図D3に基づいて、図7に示すようなノード演算子データベースD7を生成する。ノード演算子データベースD7は、複数のノード演算子レコード21を有している。ノード演算子レコード21は、ノードフィールド22と演算子フィールド23を有している。ノードフィールド22では、演算ノードN1乃至N4とレジスタR1乃至R4が記憶される。演算子フィールド23では、演算子op1乃至op4が記憶される。演算子op1乃至op4とその演算子op1乃至op4を実行する演算ノードN1乃至N4が、同じノード演算子レコード21内に記憶される。また、演算子op1乃至op4とその演算子op1乃至op4の出力データを記憶するレジスタR1乃至R4が、同じノード演算子レコード21内に記憶される。演算ノードN1乃至N4又はレジスタR1乃至R4から、同じノード演算子レコード21内に記憶された演算子op1乃至op4を検索することが可能である。

【0050】

ステップS4において、ハードウェア割付部5で、図8に示すような論理回路のデータパスD5と、このデータパスの制御情報D5を生成する。データパスD5は、演算ノードN1乃至N4として機能する演算器A1乃至A3と、レジスタ

R1乃至R4として機能する記憶素子M1とM2と、制御情報D5に基づいて伝達するデータを切り換えるセレクターC1乃至C3を有する。

【0051】

ステップS9において、図6のスケジューリングデータフロー図D3と図8のデータパスD5に基づいて、図9に示すようなノード演算器データベースD8を生成する。ノード演算器データベースD8は、複数のノード演算器レコード24を有している。ノード演算器レコード24は、演算器フィールド25、ノードフィールド26と実行ステップフィールド27を有している。演算器フィールド25では、演算器A1乃至A3と記憶素子M1、M2が記憶される。ノードフィールド22では、演算ノードN1乃至N4とレジスタR1乃至R4が記憶される。実行ステップフィールド27では、実行ステップ1乃至3が記憶される。演算器A1乃至A3と、その演算器A1乃至A3で実行する演算ノードN1乃至N4と、その演算ノードN1乃至N4が実行する実行ステップが、同じノード演算器レコード24内に記憶される。また、記憶素子M1、M2と、その記憶素子M1、M2で記憶するレジスタR1乃至R4と、そのレジスタR1乃至R4が記憶する実行ステップが、同じノード演算器レコード24内に記憶される。

【0052】

すなわち、図6のスケジューリングデータフロー図D3と図8のデータパスD5を参照しながら、図9のノード演算器データベースD8を見ると、実行ステップ1において、演算器A1が演算ノードN1として機能し、演算器A2が演算ノードN2として機能し、演算器A3は働かず、記憶素子M1がレジスタR1として機能し、記憶素子M2がレジスタR2として機能する。実行ステップ2において、演算器A1と演算器A2は働かず、演算器A3が演算ノードN3として機能し、記憶素子M1がレジスタR3として機能し、記憶素子M2は働かない。実行ステップ3において、演算器A1が演算ノードN4として機能し、演算器A2と演算器A3は働かず、記憶素子M1がレジスタR4として機能し、記憶素子M2は働かない。

【0053】

図9のノード演算器データベースD8によれば、演算ノードN1乃至N4から

、同じノード演算器レコード24内に記憶された演算器A1乃至A3を検索することが可能である。記憶素子M1、M2と実行ステップ1乃至3から同じノード演算器レコード24内に記憶されたレジスタR1乃至R4を検索することが可能である。

【0054】

ステップS10において、図7のノード演算子データベースD7と図9のノード演算器データベースD8に基づいて、図10に示すような演算子演算器データベースD9を生成する。演算子演算器データベースD9は、複数の演算子演算器レコード28を有している。演算子演算器レコード28は、演算器フィールド29、演算子フィールド30と実行ステップフィールド31を有している。演算器フィールド29では、演算器A1乃至A3と記憶素子M1、M2が記憶される。演算子フィールド30では、演算子op1乃至op4が記憶される。実行ステップフィールド31では、実行ステップ1乃至3が記憶される。演算器A1乃至A3と、その演算器A1乃至A3で実行する演算子op1乃至op4と、その演算子op1乃至op4が実行する実行ステップが、同じ演算子演算器レコード28内に記憶される。また、記憶素子M1、M2と、その記憶素子M1、M2で記憶するデータを出力する演算子op1乃至op4と、その演算子op1乃至op4が実行する実行ステップが、同じ演算子演算器レコード28内に記憶される。

【0055】

すなわち、図7のノード演算子データベースD7と図9のノード演算器データベースD8を参照しながら、図10の演算子演算器データベースD9を見ると、実行ステップ1において、演算器A1が演算子op1として機能し、演算器A2が演算子op3として機能し、演算器A3は働かず、記憶素子M1が演算子op1の出力したデータを記憶し、記憶素子M2が演算子op3の出力したデータを記憶する。実行ステップ2において、演算器A1と演算器A2は働かず、演算器A3が演算子op2として機能し、記憶素子M1が演算子op2の出力したデータを記憶し、記憶素子M2は働かない。実行ステップ3において、演算器A1が演算子op4として機能し、演算器A2と演算器A3は働かず、記憶素子M1が演算子op4の出力したデータを記憶し、記憶素子M2は働かない。

【0056】

図10の演算子演算器データベースD9によれば、演算子op1乃至op4から、同じ演算子演算器レコード28内に記憶された演算器A1乃至A3を検索することが可能である。記憶素子M1、M2と実行ステップ1乃至3から同じ演算子演算器レコード28内に記憶された演算子op1乃至op4を検索することが可能である。

【0057】

ステップS11において、動作記述D1のシミュレーションを行う。図11に示すように、入力データD13として、図4の動作記述D1の引数i1に2を代入し、引数i2に1を代入した。そして、演算子op1からの出力データD10として3を出力した。同様に、演算子op2、op3、op4からの出力データD10として3、1、4を出力した。

【0058】

次に、動作記述D1の引数i1に3を代入し、引数i2に2を代入した。そして、演算子op1、op2、op3、op4からの出力データD10として5、5、1、6を出力した。さらに、動作記述D1の引数i1に2を代入し、引数i2に3を代入した。そして、演算子op1、op2、op3、op4からの出力データD10として5、-5、-1、-4を出力した。

【0059】

なお、入力データD13と出力データD10は、図11に示すように、それぞれフィールドを構成し、入力データD13と出力データD10と演算子フィールド33からなる入力データ出力データレコード32を有するデータベースを構成してもよい。入力データ出力データレコード32を有するデータベースによれば、演算子op1乃至op4と入力データD13から、同じ入力データ出力データレコード32内に記憶された出力データD10を検索することが可能である。

【0060】

ステップS5において、RTL記述生成部6で、データパスと制御情報D5に基づいて、RTL記述D6を生成する。ステップS6において、RTL記述シミュレーション部11で、実行ステップiに、最初に実行される実行ステップ1を

設定する。

【0061】

ステップS7において、入力データD13で引数i1が2であり、引数i2が1である場合のRTL記述D6のシミュレーションを行う。図12に示すように、まず、実行ステップ1で、RTL記述D6に入力データD13を代入し、記憶素子M1、M2が記憶する記憶データD11を計算する。RTL記述D6の引数i1に2を代入し、引数i2に1を代入した。そして、実行ステップ1で、記憶素子M1の記憶データD11として3を出力し、記憶素子M2の記憶データD11として1を出力した。

【0062】

ステップS12において、図10の演算子演算器データベースD9に基づいて、計算された記憶データD11の実行ステップ1と記憶素子M1から演算子op1を検索する。また、実行ステップ1と記憶素子M2から演算子op3を検索する。

【0063】

ステップS13において、図11の検索された演算子op1の出力データD10である3と、図12の実行ステップ1で記憶素子M1の記憶データD11である3の異同を判定する。3と3で同じである。また、図11の検索された演算子op3の出力データD10である1と、図12の実行ステップ1で記憶素子M2の記憶データD11である1の異同を判定する。1と1で同じである。これらにより、ステップS14に進む。

【0064】

ステップS14において、実行ステップ1が最適化データフロー図D4の最大の実行ステップ3以上であるか否かを判定する。実行ステップ1が最大の実行ステップ3以上でないので、ステップS15に進む。ステップS15において、実行ステップ1を1つ大きく実行ステップ2にする。そして、ステップS7に進む。

【0065】

再度、ステップS7において、入力データD13で引数i1が2であり、引数

i_2 が 1 である場合で実行ステップ 2 の RTL 記述 D 6 のシミュレーションを行う。図 1 2 に示すように、実行ステップ 2 で、記憶素子 M 1 の記憶データ D 1 1 として 3 を出力した。

【0066】

ステップ S 1 2 において、図 1 0 の演算子演算器データベース D 9 に基づいて、実行ステップ 2 と記憶素子 M 1 から演算子 op 2 を検索する。

【0067】

ステップ S 1 3 において、図 1 1 の検索された演算子 op 2 の出力データ D 1 0 である 3 と、図 1 2 の実行ステップ 2 で記憶素子 M 1 の記憶データ D 1 1 である 3 の異同を判定する。3 と 3 で同じであるので、ステップ S 1 4 に進む。

【0068】

ステップ S 1 4 において、実行ステップ 2 が最適化データフロー図 D 4 の最大の実行ステップ 3 以上であるか否かを判定する。実行ステップ 2 が最大の実行ステップ 3 以上でないので、ステップ S 1 5 に進む。ステップ S 1 5 において、実行ステップ 2 を 1 つ大きく実行ステップ 3 にする。そして、ステップ S 7 に進む。

【0069】

三度、ステップ S 7 において、入力データ D 1 3 で引数 i_1 が 2 であり、引数 i_2 が 1 である場合で実行ステップ 3 の RTL 記述 D 6 のシミュレーションを行う。図 1 2 に示すように、実行ステップ 3 で、記憶素子 M 1 の記憶データ D 1 1 として 4 を出力した。

【0070】

ステップ S 1 2 において、図 1 0 の演算子演算器データベース D 9 に基づいて、実行ステップ 3 と記憶素子 M 1 から演算子 op 4 を検索する。

【0071】

ステップ S 1 3 において、図 1 1 の検索された演算子 op 4 の出力データ D 1 0 である 4 と、図 1 2 の実行ステップ 3 で記憶素子 M 1 の記憶データ D 1 1 である 4 の異同を判定する。4 と 4 で同じであるので、ステップ S 1 4 に進む。

【0072】

ステップS14において、実行ステップ3が最適化データフロー図D4の最大の実行ステップ3以上であるか否かを判定する。実行ステップ3が最大の実行ステップ3以上であるので、論理回路設計方法をストップする。

【0073】

このように、動作記述D1のミュレーション結果である演算子の出力データD10と、RTL記述D6のシミュレーション結果である記憶素子の記憶データD11が、実行ステップ毎に一致することを確認することで、動作記述からRTL記述が正確に高位合成されたことが検証できた。

【0074】

(実施例2)

RTL記述D6のシミュレーションを、1実行ステップ毎にデータを比較をするのではなく、RTL記述D6の全実行ステップの実行後に動作記述D1とRTL記述の比較検証を行う場合について説明する。入力データD13は、引数i1が3であり、引数i2が2であるとする。実施例1とステップS1乃至S5、S8乃至S11は同じである。ステップS12、S13、S16、S17の実施に先駆けて、ステップS6、S7、S14とS15を実施する。ステップS7、S14とS15でループを構成する。実行ステップ1から最大の実行ステップ3までの実行ステップiについて、入力データD13で引数i1が3であり、引数i2が2である場合のRTL記述D6のシミュレーションを行う。図12に示すように、まず、実行ステップ1で、記憶素子M1の記憶データD11として5を出力し、記憶素子M2の記憶データD11として1を出力した。次に、実行ステップ2で、記憶素子M1の記憶データD11として5を出力した。最後に、実行ステップ3で、記憶素子M1の記憶データD11として6を出力した。

【0075】

ステップS12において、図10の演算子演算器データベースD9に基づいて、計算された記憶データD11の実行ステップ1と記憶素子M1から演算子op1を検索する。また、実行ステップ1と記憶素子M2から演算子op3を検索する。実行ステップ2と記憶素子M1から演算子op2を検索する。実行ステップ3と記憶素子M1から演算子op4を検索する。

【0076】

ステップS13において、図11の検索された演算子op1の出力データD10である5と、図12の実行ステップ1で記憶素子M1の記憶データD11である5の異同を判定する。5と5で同じである。また、図11の検索された演算子op3の出力データD10である1と、図12の実行ステップ1で記憶素子M2の記憶データD11である1の異同を判定する。1と1で同じである。図11の検索された演算子op2の出力データD10である5と、図12の実行ステップ2で記憶素子M1の記憶データD11である5の異同を判定する。5と5で同じである。図11の検索された演算子op4の出力データD10である6と、図12の実行ステップ3で記憶素子M1の記憶データD11である6の異同を判定する。6と6で同じである。これらの判定により、論理回路設計方法をストップする。

【0077】

このように、動作記述D1のミュレーション結果である演算子の出力データD10と、RTL記述D6のシミュレーション結果である記憶素子の記憶データD11を、全実行ステップにわたって一致することを確認することで、動作記述からRTL記述が正確に高位合成されたことが検証できた。

【0078】

(実施例3)

RTL記述D6に不具合がある場合について説明する。RTL記述D6のシミュレーションは、実施例1と同様に、RTL記述D6の全実行ステップの実行後に動作記述D1とRTL記述D6の比較検証を行った。入力データD13は、引数i1が2であり、引数i2が3であるとする。実施例1とステップS1乃至S5、S8乃至S11は同じである。実施例2と同様に、ステップS12、S13、S16、S17の実施に先駆けて、ステップS6、S7、S14とS15を実施する。図12に示すように、まず、実行ステップ1で、記憶素子M1の記憶データD11として5を出力し、記憶素子M2の記憶データD11として-1を出力した。次に、実行ステップ2で、記憶素子M1の記憶データD11として1275を出力した。最後に、実行ステップ3で、記憶素子M1の記憶データD11として12

76を出力した。

【0079】

ステップS12において、図10の演算子演算器データベースD9に基づいて、計算された記憶データD11の実行ステップ1と記憶素子M1から演算子op1を検索する。また、実行ステップ1と記憶素子M2から演算子op3を検索する。実行ステップ2と記憶素子M1から演算子op2を検索する。実行ステップ3と記憶素子M1から演算子op4を検索する。

【0080】

ステップS13において、図11の検索された演算子op1の出力データD10である5と、図12の実行ステップ1で記憶素子M1の記憶データD11である5の異同を判定する。5と5で同じである。また、図11の検索された演算子op3の出力データD10である-1と、図12の実行ステップ1で記憶素子M2の記憶データD11である-1の異同を判定する。-1と-1で同じである。図11の検索された演算子op2の出力データD10である-5と、図12の実行ステップ2で記憶素子M1の記憶データD11である1275の異同を判定する。-5と1275は異なる。図11の検索された演算子op4の出力データD10である-4と、図12の実行ステップ3で記憶素子M1の記憶データD11である1276の異同を判定する。-4と1276は異なる。異なる判定が生じたのでステップS16に進む。検索された演算子op2とop4で異なる判定が生じた。

【0081】

ステップS16において、図10の演算子演算器データベースD9に基づいて、異なる判定が生じた演算子op2から演算器A3を検索する。同様に、異なる判定が生じた演算子op4から演算器A1を検索する。

【0082】

ステップS17において、論理回路設計装置1のオペレータが、検索された演算器A3とA1に基づいて、RTL記述D6をデバックする。なお、演算器A3は演算器A1より早い実行ステップ2で不具合を発生させている。演算器A3での不具合により、演算器A1の不具合が発生する場合も考えられる。このような

場合は、まず、演算器A3に関してデバックを行えばよい。そして、ステップS5に戻る。ステップS5に戻ることで、デバックされたRTL記述D6の記憶素子の記憶データD11の値が、動作記述D1の演算子の出力データD10に一致するまで、オペレータはデバックを繰り返すことができる。

【0083】

このように、動作記述D1とRTL記述D6の比較検証をすることにより、不具合が生じた場合には、動作記述D1とRTL記述D6の両者での不具合箇所が特定できる。このことにより、RTL記述D6の修正が容易に行え、設計にかかる時間を大幅に短縮することができる。

【0084】

(実施例4)

実施例4でも、論理回路設計装置1で論理回路設計方法を用いた。実施例4では最適化を実施した場合について説明する。実施例4の論理回路設計方法は、図3のステップS1において、図13に示すような複数の演算子op1乃至op3を有する動作記述D1から、演算子op1乃至op3を実行する演算ノードN1乃至N3を実行順に配列した未加工データフロー図D2に変換する。

【0085】

ステップS2において、図14に示すように、未加工データフロー図D2に実行ステップ1乃至3を割り当てる。入力データD13の引数i1とi2で、演算ノードN1において、和算の演算子op1を実行する。演算ノードN1は、実行ステップ1で実行するという時間情報が添付される。演算ノードN1からの出力データを記憶するレジスタR1を実行ステップ1の実行後に挿入する。

【0086】

演算ノードN1の演算子op1の出力と引数i3で、演算ノードN2において、和算の演算子op2を実行する。演算ノードN2は、実行ステップ2で実行するという時間情報が添付される。演算ノードN2からの出力データを記憶するレジスタR2を実行ステップ2の実行後に挿入する。

【0087】

演算ノードN2の演算子op2の出力と引数i4で、演算ノードN3において

、和算の演算子 $o_p 3$ を実行する。演算ノード N 3 は、実行ステップ 3 で実行するという時間情報が添付される。演算ノード N 3 からの出力データを記憶するレジスタ R 3 を実行ステップ 3 の実行後に挿入する。

【0088】

スケジューリング部 3 は、このように加工した未加工データフロー図 D 2 を、図 14 のスケジューリングデータフロー図 D 3 として出力する。なお、後述するステップ S 8 の理解が容易になるように、ステップ S 8 の一部を先行して実施する。図 13 の動作記述 D 1 と図 14 のスケジューリングデータフロー図 D 3 に基づいて、図 15 に示すようなノード演算子データベース D 7 を生成する。ノードフィールド 2 2 では、演算ノード N 1 乃至 N 3 とレジスタ R 1 乃至 R 3 が記憶される。演算子フィールド 2 3 では、演算子 $o_p 1$ 乃至 $o_p 3$ が記憶される。演算子 $o_p 1$ 乃至 $o_p 3$ とその演算子 $o_p 1$ 乃至 $o_p 3$ を実行する演算ノード N 1 乃至 N 3 が、同じノード演算子レコード 2 1 内に記憶される。また、演算子 $o_p 1$ 乃至 $o_p 3$ とその演算子 $o_p 1$ 乃至 $o_p 3$ の出力データを記憶するレジスタ R 1 乃至 R 3 が、同じノード演算子レコード 2 1 内に記憶される。

【0089】

ステップ S 3 において、図 14 のスケジューリングデータフロー図 D 3 を、図 16 に示すように最適化し、最適化データフロー図 D 4 を生成した。直列に処理される和算を並列に処理するように組み替えた。演算ノード N2、N3 とレジスタ R2 が消去されて、代わりに演算ノード N4、N5 とレジスタ R4 が生成された。予想される全実行時間が、実行ステップで 3 実行ステップから 2 実行ステップに短縮された。

【0090】

ステップ S 8 において、図 13 の動作記述 D 1 と図 14 のスケジューリングデータフロー図 D 3 と図 16 の最適化データフロー図 D 4 に基づいて、まず、図 15 のノード演算子データベース D 7 を生成し、さらに、図 17 に示すような、ノード演算子データベース D 7 を生成する。

【0091】

ノードフィールド 2 2 では、演算ノード N 1 乃至 N 5 とレジスタ R 1 乃至 R 4

が記憶される。演算子フィールド23では、演算子op1乃至op3が記憶される。演算子op1乃至op3とその演算子op1乃至op3を実行する演算ノードN1乃至N4が、同じノード演算子レコード21内に記憶される。なお、演算子op2とop3に直接対応する演算ノードN2とN3は、図16の最適化データフロー図D4では存在しない。そこで、演算子op2とop3の実行に影響を受ける演算ノードに、演算ノードN4とN5を設定する。演算子op2とop3とその演算子op2とop3の実行に影響を受ける演算ノードN4が、同じノード演算子レコード21内に記憶される。演算子op2とop3とその演算子op2とop3の実行に影響を受ける演算ノードN5が、同じノード演算子レコード21内に記憶される。

【0092】

また、演算ノードN1と、演算ノードN1に対応するレジスタR1は、演算子op1に関係付けられている。演算ノードN4と、演算ノードN4に対応するレジスタR4は、演算子op2、op3に関係付けられている。演算ノードN5と、演算ノードN5に対応するレジスタR3は、演算子op2、op3に関係付けられている。

【0093】

ステップS4において、図18に示すような論理回路のデータパスD5と、このデータパスの制御情報D5を生成する。データパスD5は、演算ノードN1、N5として機能する演算器A1と、演算ノードN4として機能する演算器A2と、レジスタR1、R3として機能する記憶素子M1と、レジスタR4として機能する記憶素子M2と、制御情報D5に基づいて伝達するデータを切り換えるセレクターC1、C2を有する。

【0094】

ステップS9において、図16の最適化データフロー図D4と図18のデータパスD5に基づいて、図19に示すようなノード演算器データベースD8を生成する。図19のノード演算器データベースD8より、実行ステップ1において、演算器A1が演算ノードN1として機能し、演算器A2が演算ノードN4として機能し、記憶素子M1がレジスタR1として機能し、記憶素子M2がレジスタR

4として機能する。実行ステップ2において、演算器A1が演算ノードN5として機能し、記憶素子M1がレジスタR3として機能する。

【0095】

ステップS10において、図17のノード演算子データベースD7と図19のノード演算器データベースD8に基づいて、図20に示すような演算子演算器データベースD9を生成する。図20の演算子演算器データベースD9を見ると、実行ステップ1において、演算器A1が演算子op1として機能し、演算器A2が演算子op2、op3として機能し、記憶素子M1が演算子op1の出力したデータを記憶し、記憶素子M2が演算子op2、op3の出力したデータを記憶する。実行ステップ2において、演算器A1が演算子op2、op3として機能し、記憶素子M1が演算子op3の出力したデータを記憶する。

【0096】

図20の演算子演算器データベースD9によれば、演算子op1乃至op3から、同じ演算子演算器レコード28内に記憶された演算器A1、A2を検索することが可能である。記憶素子M1、M2と実行ステップ1、2から同じ演算子演算器レコード28内に記憶された演算子op1乃至op3を検索することが可能である。例えば、記憶素子M1と実行ステップ1から演算子op1を検索できる。そして、演算子op1から演算器A1を検索できる。また、記憶素子M2と実行ステップ1から演算子op2、op3を検索できる。演算子op2、op3と実行ステップ1から演算器A2を検索できる。

【0097】

このように、動作記述D1からRTL記述D6を生成する際に、動作記述D1とRTL記述D6の両者を対応づける演算子演算器データベースD9が生成できる。このことにより、動作記述D1とRTL記述D6の比較検証をすることができる。この比較検証で不具合が生じた場合には、動作記述D1、RTL記述D6の両者での不具合箇所が特定できる。動作記述D1やRTL記述D6の修正が容易に行える。

【0098】

【発明の効果】

以上説明したように、本発明によれば、動作記述から変換されたRTL記述で

の論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供できる。

【0099】

本発明によれば、コンピュータが動作記述から変換されたRTL記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計プログラムを提供せきる。

【0100】

本発明によれば、動作記述から変換されたRTL記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計装置を提供できる。

【図面の簡単な説明】

【図1】

本発明の実施の形態に係る論理回路設計装置の構成図である。

【図2】

本発明の実施の形態に係る論理回路設計装置のデータフロー図である。

【図3】

本発明の実施の形態に係る論理回路設計方法のフローチャートである。

【図4】

実施例1の動作記述を表す図である。

【図5】

実施例1の未加工のDFGを表す図である。

【図6】

実施例1のスケジューリング後のDFGを表す図である。

【図7】

実施例1のノード演算子データベースのデータ構造を表す図である。

【図8】

実施例1のデータパスを表す図である。

【図9】

実施例1のノード演算器データベースのデータ構造を表す図である。

【図10】

実施例1の演算子演算器データベースのデータ構造を表す図である。

【図11】

実施例1の動作記述のシミュレーションの結果を表す図である。

【図12】

実施例1のRTL記述のシミュレーションの結果を表す図である。

【図13】

実施例4の動作記述を表す図である。

【図14】

実施例4のスケジューリング後のDFGを表す図である。

【図15】

実施例4のノード演算子データベースの最適化前のデータ構造を表す図である

【図16】

実施例4の最適化後のDFGを表す図である。

【図17】

実施例4のノード演算子データベースの最適化後のデータ構造を表す図である

【図18】

実施例4のデータパスを表す図である。

【図19】

実施例4のノード演算器データベースのデータ構造を表す図である。

【図20】

実施例4の演算子演算器データベースのデータ構造を表す図である。

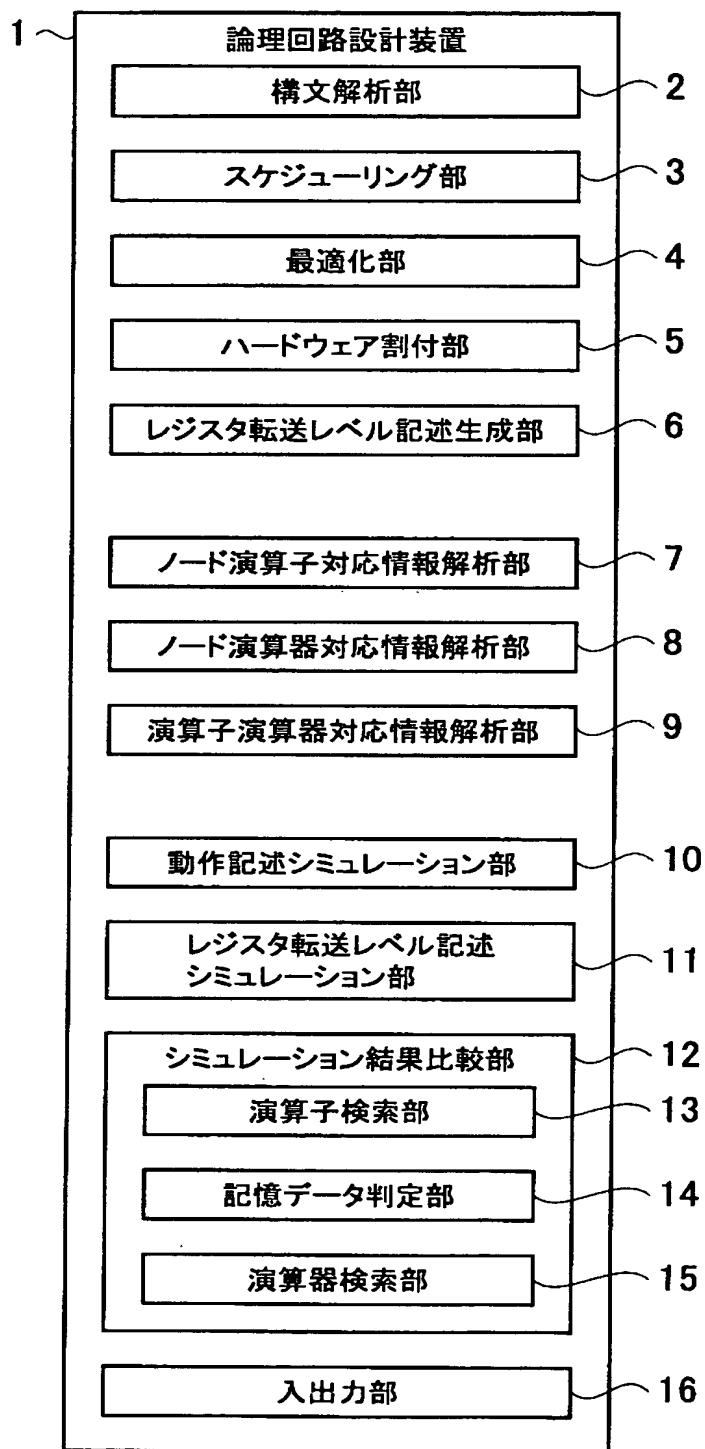
【符号の説明】

- 1 論理回路設計装置
- 2 構文解析部
- 3 スケジューリング部
- 4 最適化部
- 5 ハードウェア割付部

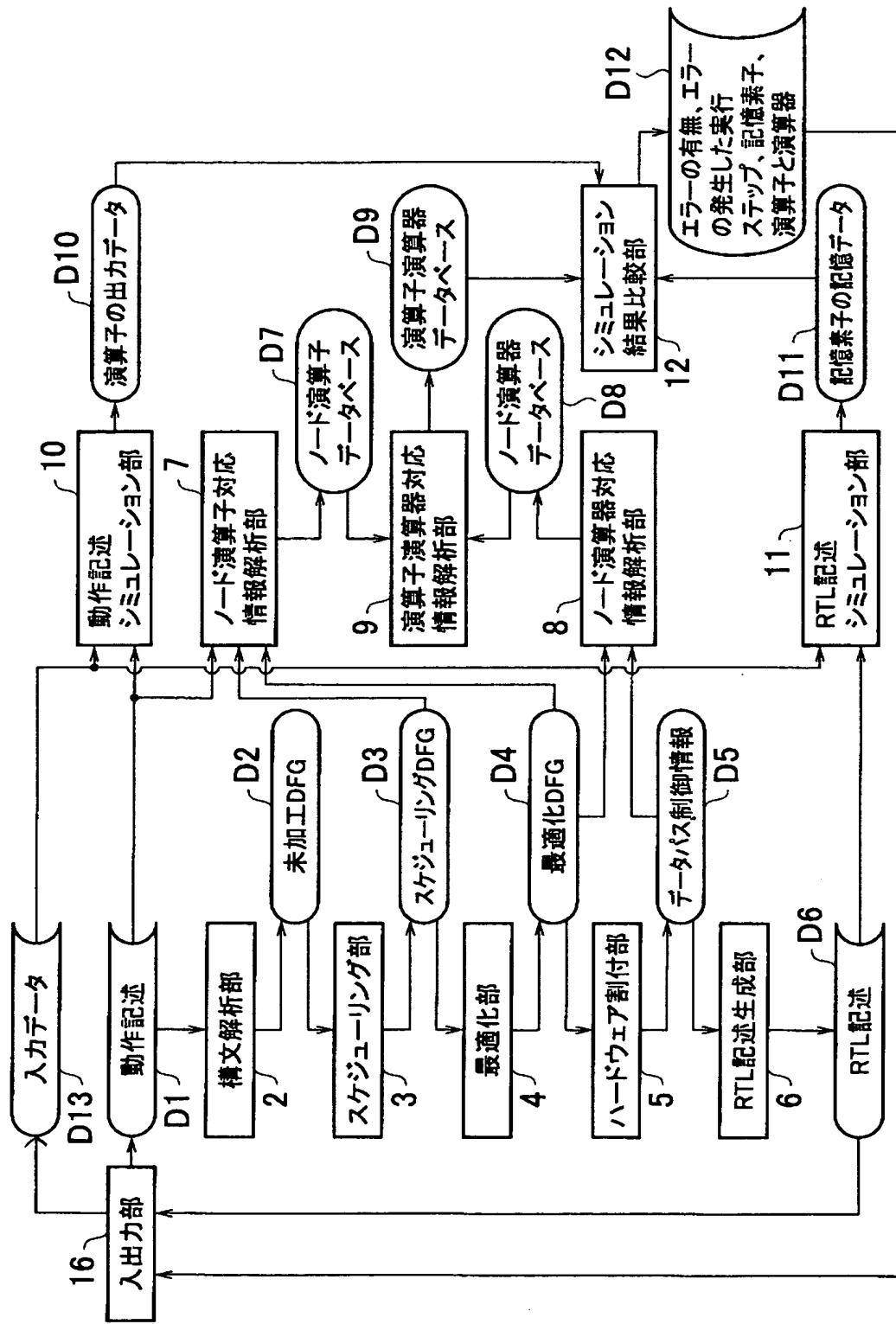
- 6 レジスタ転送レベル (R T L) 記述生成部
- 7 ノード演算子対応情報解析部
- 8 ノード演算器対応情報解析部
- 9 演算子演算器対応情報解析部
- 10 動作記述シミュレーション部
- 11 R T L記述シミュレーション部
- 12 シミュレーション結果比較部
- 13 演算子検索部
- 14 記憶データ判定部
- 15 演算器検索部
- 16 入出力部
- 21 ノード演算子レコード
- 22 ノードフィールド
- 23 演算子フィールド
- 24 ノード演算器レコード
- 25 演算器フィールド
- 26 ノードフィールド
- 27 実行ステップフィールド
- 28 演算子演算器レコード
- 29 演算器フィールド
- 30 演算子フィールド
- 31 実行ステップフィールド
- 32 入力データ出力データレコード
- 33 演算子フィールド
- 34 入力データ記憶データレコード
- 35 記憶素子フィールド
- 36 実行ステップフィールド

【書類名】 図面

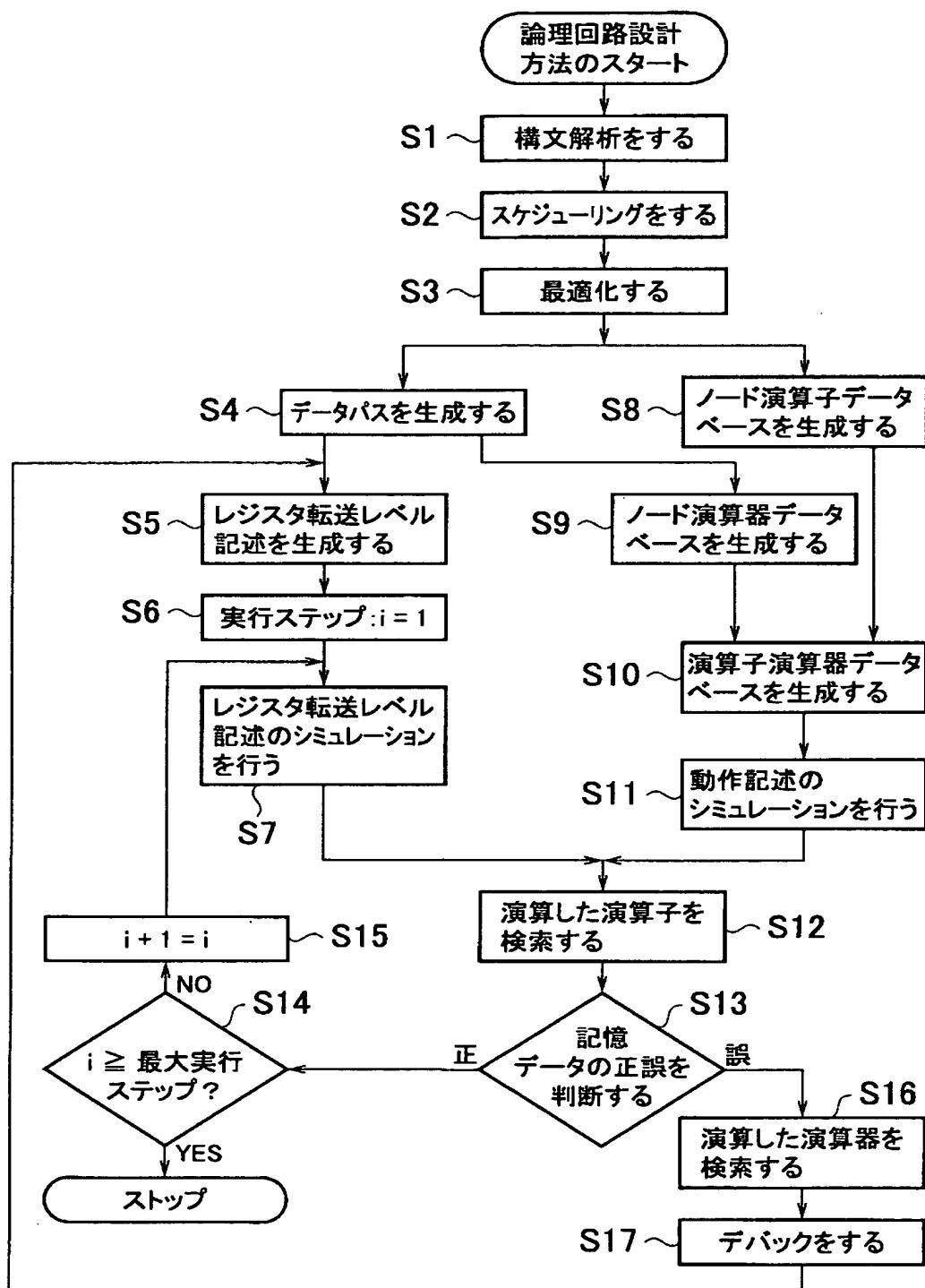
【図1】



【図2】



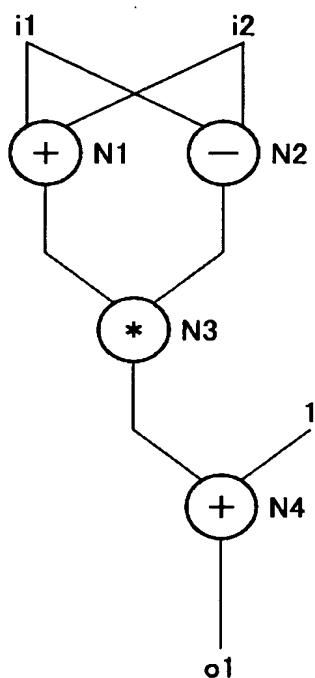
【図3】



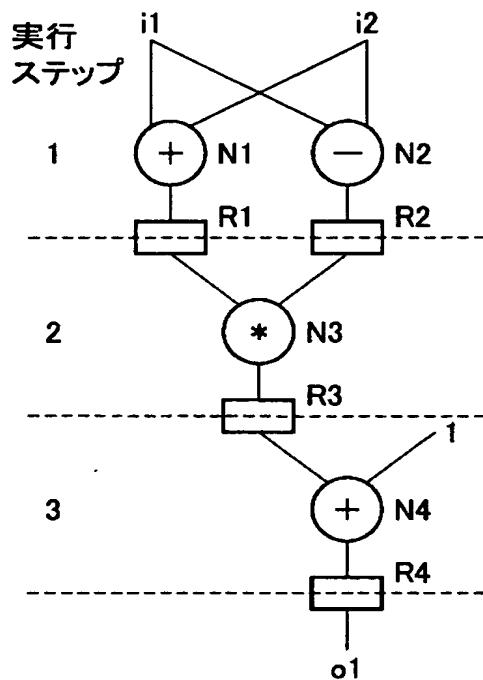
【図4】

```
void calc(int i1, int i2, int &o1)
{
    o1 = (i1 + i2) * (i1 - i2) + 1;
}          op1   op2   op3   op4
```

【図5】



【図6】



【図7】

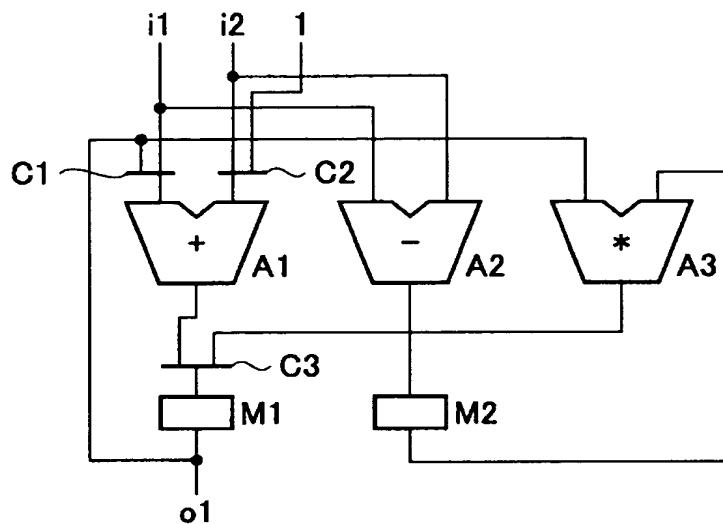
21 {

ノード	演算子
N1	op1
N2	op3
N3	op2
N4	op4
R1	op1
R2	op3
R3	op2
R4	op4

D7

22 { 23 }

【図8】



【図9】

24 {

演算器	実行ステップ			D8
	1	2	3	
A1	N1		N4	27
A2	N2			
A3		N3		
M1	R1	R3	R4	
M2	R2			

25 { 26 }

【図10】

演算器	実行ステップ			D9	31
	1	2	3		
28 {	A1	op1		op4	
	A2	op3			
	A3		op2		
	M1	op1	op2	op4	
	M2	op3			

29 30

【図11】

(i1, i2)	演算子				33
	op1	op2	op3	op4	
(2, 1)	3	3	1	4	
(3, 2)	5	5	1	6	
(2, 3)	5	-5	-1	-4	

D13 D10

【図12】

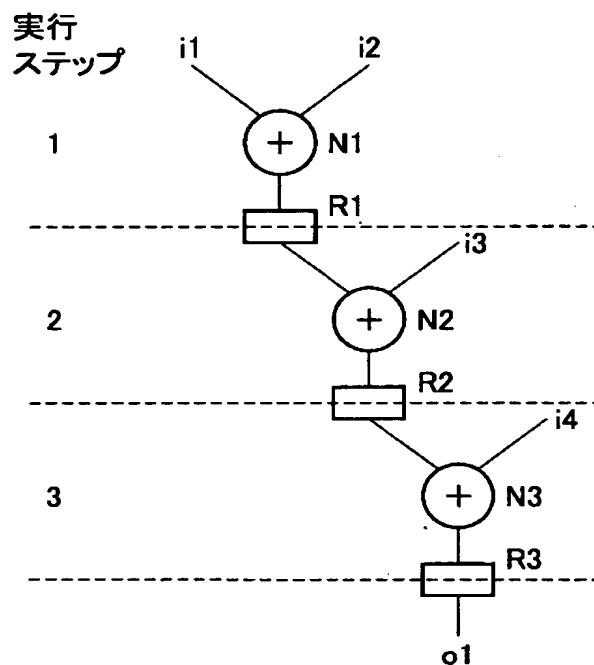
(i1, i2)	実行ステップ				36
		1	2	3	
(2, 1)	M1	3	3	4	
	M2	1			
(3, 2)	M1	5	5	6	
	M2	1			
(2, 3)	M1	5	1275	1276	
	M2	-1			

D13 35 D11

【図13】

```
void calc(int i1, int i2, int i3, int i4, int &o1)
{
    o1 = i1 + i2 + i3 + i4;
}          op1 op2 op3
```

【図14】



【図15】

21 {

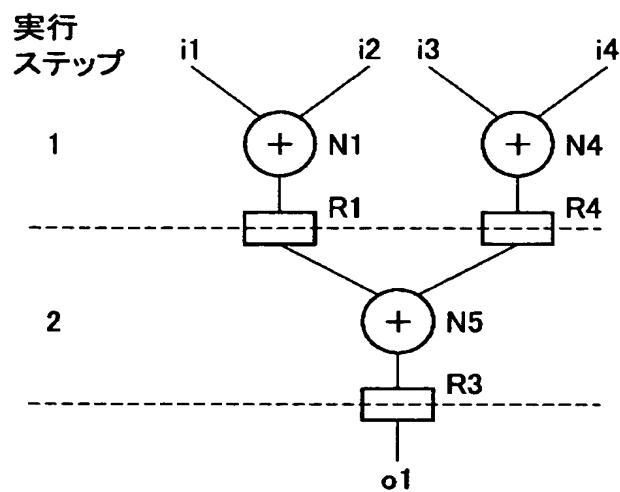
ノード	演算子
N1	op1
N2	op2
N3	op3
R1	op1
R2	op2
R3	op3

22

23

~D7

【図16】

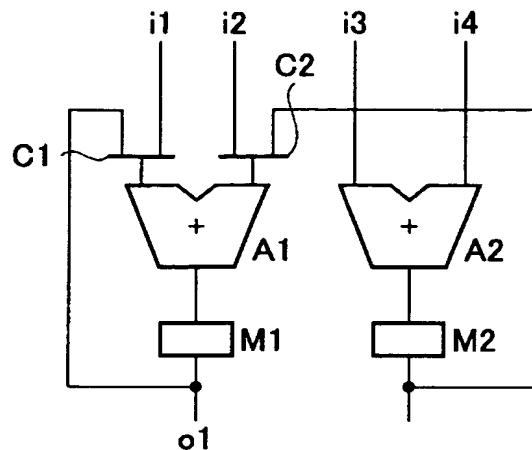


【図17】

ノード	演算子	D7
N1	op1	
N2	op2	
N3	op3	
N4	op2, op3	
N5	op2, op3	
R1	op1	
R2	op2	
R3	op2, op3	
R4	op2, op3	

22 23

【図18】



【図19】

24 {

演算器	実行ステップ		D8
	1	2	
A1	N1	N5	27
A2	N4		
M1	R1	R3	
M2	R4		

25 { 26 }

【図20】

28 {

演算器	実行ステップ		D9
	1	2	
A1	op1	op2, op3	31
A2	op2, op3		
M1	op1	op2, op3	
M2	op2, op3		

29 { 30 }

【書類名】 要約書

【要約】

【課題】 動作記述から変換されたR T L記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供する。

【解決手段】 論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換し、データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する。演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成する。演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する。

【選択図】 図3

出願人履歴情報

識別番号 [000003078]

1. 変更年月日 2001年 7月 2日

[変更理由] 住所変更

住 所 東京都港区芝浦一丁目1番1号

氏 名 株式会社東芝